attached to *x* in FIG. 10. If *x* forwards one copy of the data frame to *w* and one to *v*, *u'* may receive two copies of the same data frame, one from *w* and one from *v*. The STAR Bridge Protocol avoids this by allowing only *w*, the agent bridge of *s*, to forward the frame using, in this case, an enhanced forwarding path. Since the agent bridge is unique, at most one copy of the frame may be sent to the destination.

V.B Frame Dropping Problem

Another problem is due to the existence of old bridges. Although the STAR bridges know how to forward a frame on an enhanced forwarding path after knowing the location of the destination, old bridges don't. In some cases, old bridges will drop a frame trying to pass through. In FIG. 10, bridges *w*, *x*, and *v'* are on the same branch and there is a crosslink between *v'* and *z*. Let *z* be the designated bridge of end station *s1*. As *x* and *z* are on different branches, in the FD of *x*, it marks the root port as the forwarding port of *s1*. However, if *w* wants to send a frame to *s1* and finds out the shortest path to *z* is going through *v'*, it sends the frame with destination address *s1* to *x*. The frame will be dropped by *x* since it is coming from the forwarding direction. To fix this problem, we will encapsulate the normal data frame with an appropriate proxy destination address so that *x* will forward the frame towards *v'*, but not other directions as in Section I.E. An old bridge may drop a data frame only if the data frame is being forwarded on a tree path. It also implies that the frame is trying to go from one STAR bridge to another through an old bridge tree path between them. If the destination address of a frame is the MAC address of the next hop bridge, all the old bridges along the tree path will forward the frame to the next hop as desired. In the present example, *w* encapsulates the frame with the MAC address of *v'* as a proxy destination address, such that *x*, upon receiving the encapsulated frame, will forward the frame to *v'* without dropping it. In general, when the next hop STAR bridge is not a tree neighbor, the sender STAR bridge will encapsulate the data frame. Since the encapsulated data frames are encapsulated using the sender STAR bridge MAC address as the source address, old bridges will learn the forwarding directions to the sender STAR bridge. In this respect, no additional control message is needed to enable old bridges to learn the forwarding direction to any STAR bridge.

A BA Table is used to keep the MAC addresses of all distant STAR neighbors. Since STAR bridge *n* puts its MAC address as the source address of the DVMyInfo and DVOurInfo frames it sends, we don't need an extra SBPDU frame to fill this table.

V.D Redundant Traversal Problem

The encapsulation approach described in Section V.B may prevent a designated old bridge from identifying a normal data frame that is destined to it. Referring to the configuration in FIG. 10, let end station *s* be attached to the old bridge *x*. According to the protocol, STAR bridge *w* will declare itself to be the agent bridge of *s*, so that all frames destined for *s* will be forwarded as though they were destined for *w*. Suppose that *s1* wants to send a frame to *s*. STAR bridge *z*, which is *db(s1)*, will forward the frame via a crosslink to *v'* then. Since the next hop to a STAR bridge is *w* and it is on a tree path, *v'* will encapsulate the frame with *w* as a proxy destination. When old bridge *x* receives the frame, it will think that it is not a frame addressed to itself and forward it to the proxy destination *w*. When *w*

receives it, it will strip off the encapsulation header and send the normal data frame back to *x*. Then, *x* can identify the frame and send it to the destination end station. Therefore, the data frame traverses a redundant path from the designated bridge to the agent bridge and back to the designated bridge. In the case where the agent bridge of source end station, *ab(s)*, and the agent bridge of the destination end station, *ab(t)*, are on different branches, we will show later in Section VII.C. that the total distance traversed is still no worse than the corresponding tree path. However, when *ab(s)* and *ab(t)* are on the same branch, the total distance traversed may be longer than the tree path. We avoid this by not encapsulating the normal data frame in this situation. To let the next hop STAR bridge know whether a frame is intended to be forwarded on an enhanced forwarding path or a tree path, the agent bridge always encapsulates a frame that is going to be forwarded on an enhanced forwarding path.

V.E Procedure for STAR Forwarding Process

To avoid unnecessary frame dropping, STAR bridge *n* encapsulates a normal data frame as discussed in Section I.E. In the encapsulated data frame, the MAC address of *n* is used as the source address and the MAC address of the next hop STAR bridge is used as the destination address. Given a data frame *fr*, if it is a normal data frame, *src(fr)* and *dst(fr)* is the source end station address and the destination end station address respectively. If *fr* is an encapsulated data frame, it must have been encapsulated by a STAR bridge whose MAC address is *src(fr)*, and *dst(fr)* is the address of the intended STAR bridge recipient. We let *encap(fr, srcbridge, dstbridge)* as the encapsulated data frame of a normal data frame *fr* where *srcbridge* and *dstbridge* are respectively the source and destination addresses associated with the encapsulated frame. We use *uncap(fr)* to represent the normal data frame that an encapsulated data frame *fr* is carrying in its payload. Pseudocode 7 presents the FD_Search procedure for finding information from the FD. Pseudocode 8 presents the ESL_Search procedure for finding information from the ESL Table. In all the pseudocodes of this section, *self* is the STAR bridge executing the process, *p* is the receiving port of the data frame, *s* is the source end station, *t* is the destination end station, and *pld* is the payload portion of a normal data frame. FIG. 19 and FIG. 20 show the flow-charts corresponding to Pseudocode 7 and Pseudocode 8 respectively.

In Pseudocode 8 (ESL_Search_Proc(*s, t, pld, p*)), when the agent bridge of end station *t* is not known (Case 8.1). It is an error case and the frame should be dropped. This is an error because DF_STAR_Forwarding_Proc calls ESL_Search_Proc only when *self* knows that *ab(t)* is defined (Case 9.2) or *ab(t)* is known by some STAR bridge that encapsulates the frame (Case 9.1). When *self* itself is the agent bridge of *t* (Case 8.2), it sends the frame to the forwarding port leading to *t*. In Case 8.3, the agent bridge is another STAR bridge. In this case, the BF Table should give the forwarding port and the next hop information. If not, there is error and the frame is dropped.

Pseudocode 9 is a complete new forwarding procedure of bridge *self*. FIG. 22 shows the flow-chart corresponding to Pseudocode 9. When the data frame is encapsulated, it must be sent from another STAR bridge *n'* , which has the information of *ab(t)* in its ESL Table and BF Table. Therefore, *self*

should search from its ESL Table to forward the frame. On the other hand, when the data frame is not encapsulated, there are several situations. If *ab(t)* is unknown (Case 9.2a), of course, *self* should try to look at the FD. Both Case 9.2b and Case 9.2c are the cases where *ab(t)* is known. In Case 9.2b, *self* is the agent bridge of the source end station. In this case, when *ab(t)* and ab(s) are on the same branch, the normal data frame should not be encapsulated as explained in Section V.C. We don't search the ESL Table in Case 9.2c because of the frame duplication issue discussed in Section V.A. Only a tree path can be used to forward the data frame in this case. FIG. 8 and FIG. 21 show the procedures for processing a data frame in accordance with the STAR Bridge Protocol and the IEEE 802.1D Spanning Tree Bridge Protocol respectively.

PROCEDURE: FD_Search_Proc(*s, t, pld, p*), also see Fig. 19

```
Begin
    if f(self, t) is not found          /* Case 7.1: End station t is unknown */
        send data frame <s, t, pld> on all tree ports except p
    else if f(self, t) = p              /* Case 7.2: Forwarding port is the coming port */
        drop the frame
    else
        send data frame <s, t, pld> on f(self, t)
    endif
end
```

Pseudocode 7: FD_Search_Proc

PROCEDURE: ESL_Search_Proc(*s, t, pld, p*), also see Fig. 20

```
Begin
    if ESL(self, t) is not found                    /* Case 8.1: error */
        drop the frame
    Else if ESL(self, t) = self                     /* Case 8.2: ab(t) = self */
        Send data frame <s,t,pld> on f(self,t)
    Else                                            /* Case 8.3: ab(t) is known and ab(t) ≠ self */
        a := ab(t)
        if F(self, a) is not found or next(self, a) is not found    /* Case 8.3a: error */
            drop the frame
        else                                        /* Case 8.3b: encapsulation necessary */
            send data frame encap(<s, t, pld>, addr(self), addr(next(self, a)))
            on F(self, a)
        endif
    endif
end
```

Pseudocode 8: ESL_Search_Proc